Placing Servers in Overlay Networks

Sherlia Shi Jonathan Turner
Department of Computer Science
Washington University in St. Louis
One Brookings Dr.
Campus Box 1045
St. Louis, MO 63130
{*sherlia, jst*}*@cs.wustl.edu*

ABSTRACT

Overlay networks are becoming a popular vehicle for deploying advanced network services in the Internet. Typically, overlay networks are implemented by deploying *service nodes* at suitably chosen sites in the network. The number of distinct service nodes has a big influence on the operational cost of an overlay network; meanwhile, the distance between service nodes and end users has a big influence on the quality of the service that can be provided through the commodity Internet. In this paper, we study the problem of how to optimally place service nodes in a network, balancing the need to minimize the number of nodes, while limiting the distance between users and service nodes. We show that the design problem is NP-hard and study the performance of heuristic algorithms using simulations. For single domain, our algorithms produce results that are within a few percent of an easily computed lower bound. For multi-domain networks, the performance ranges from close to the optimal to roughly twice the optimal.

Keywords – overlay networks, network planning, quality-of-service, set cover

1 INTRODUCTION

As the explosive growth of the Internet continues, service providers are pushing more network functions towards the network edges to reduce client access latency and achieve better scalability. This distributed server model is often referred to as overlay network, since servers form an overlay of unicast connections to cooperate and communicate over the general Internet infrastructure. Content providers, such as Akamai [19] and iBeam [8] are among the first to deploy an overlay network of content distribution servers. Newer value-added services such as Active Networks, also adopt the overlay network approach.

In an overlay network, the communication channels between servers and clients and among server themselves are through the commodity Internet. While the server-to-server paths can be explicitly provisioned to ensure the available bandwidth and latency requirements, it is generally not cost-efficient to implement the resource management and reservation functions on the more numerous client access paths. Consequently, the quality of the services are determined largely by the network locations of the deployed servers. The current Internet has thousands of ISPs. In order to serve more clients, servers are placed strategically at the peering points of the networks to interconnect with as many ISPs as possible. However, operating and maintaining these distributed servers represent a major cost for service providers, limiting the number of servers that can be deployed. Additionally, as the number of servers grows large, the cost of interconnecting these servers for data and state synchronization also increases.

In this paper, we attempt to answer the following questions: *Given multiple networks and their estimated service parameters, how many servers are needed and where to place them, so that an overlay service provider can ensure the desired service quality to all its clients.* We envision that this imposition of the service quality constraints on server to client paths is essential for the newer network services to achieve better service quality in order to attract and retain customers. The measure of service quality can vary from application to application: it can be delay for real-time applications, or bandwidth for content distribution applications, or a combination of both. The connection from a client to its designated server node can stay within an ISP domain or may cross multiple ISP domains. Within an ISP network, the service provider can estimate these service quality parameters for a given client to a potential server location based on the client's network access technology and the capacities of the internal routing paths. Across the ISP domains, such estimation is also possible if the peering path between networks is explicitly indicated or if both networks guarantee a service level agreement from which we can infer the service parameter. The details of such estimation mechanisms are beyond the scope of this paper, hereby we assume that we can decide in advance whether or not placing a server at a specific location can provide a given client with the desired level of service quality. In this paper, we will simply use the network distance between a client and a server as the service parameter; however, our methods can

apply to any generic metrics.

To answer the above question, we transform the placement problem to the set cover problem [1] and solve it using both linear programming (LP) relaxation and greedy heuristics. An instance of the set cover problem is that given a base set of elements and a family of sets that are subsets of this base set, find the minimum number of sets such that their union includes all elements in the base set. The server placement maps to the set cover problem as follows: an element corresponds to the network location of an edge router, which represents the aggregation of regional clients in an ISP network; The base element set contains all the network locations of edge routers; A set represents a potential server placement at one of the network locations; Each set includes all the network locations that are within the service range from the server location represented by the same set. By solving the set cover problem, we find the minimum number of servers and their locations, that will cover all clients within the service range. In this paper, we will only consider the uncapacitated version of the set cover problem, where the servers do not have capacity limits and can serve as many clients as possible. We think this uncapacitated version is adequate since it is typically cheaper to buy more bandwidth at one location than to install a separate server. The set cover problem is NP-Hard [11] and has worst case approximation ratio of $O(\log n)$ [4, 15]. We introduce a rounding technique to solve the integer-programming formulation of the set cover problem based on the linear programming (LP) relaxation methods. The super-optimality of the LP problem provides a lower bound to the IP formulation of the set cover problem. Using simulation, we show that this rounding technique approaches the lower bound very closely; in fact, it reaches the lower bound for a number of network configurations. Meanwhile, the greedy heuristic also provides good performance in all instances with significantly less computation complexity.

We also vary the problem to allow *primary* and *backup* servers. A primary server provides the guaranteed service to clients, while a backup server is allowed to provide a reduced level of service quality and functions only when the primary fails. Since more servers are qualified to perform as backups, we can achieve better service reliability with only a small increase in the required number of servers.

One important aspect of our study is the network modeling used in our simulation. Existing network modeling tools, such as GT-ITM [21] and Tiers [3], can generate hierarchical network graphs with probabilistic network interconnections, however, they do not explicitly model the geographical locations of the network elements. In our model, we consider the potential of co-located servers which can access multiple networks from the same geographical location; this mirrors the behavior of co-location service providers in the current Internet. With co-location, if two nodes of different networks are within a geographical vicinity, a server installed at this location can service clients, who are within the service range, in both networks. We show that these co-locations can greatly reduce the number of required servers, since they can avoid detours through the network peering points by providing shortcuts from one network to another.

The rest of the paper is organized as follows: in Section 2, we discuss some of the related work; we describe the two network models used in our simulation in Section 3; Section 4 introduce our methods using LP-relaxation and the greedy heuristic; We present simulation results in Section 5 and conclude in Section 6.

2 RELATED WORK

Our formulation of the overlay network placement problem is related to two other well-studied problems: the facility location problem and the k-median problem. The facility location problem minimizes the joint cost of server installation and the cost of connecting each client to its designated server. This problem has been applied to designing and placing network concentrators. The k-median problem minimizes the cost of connections between clients and servers under the constraint that no more than k servers can be used. Both problems are NP-Hard. The best known approximation algorithms can achieve constant ratio [5, 9, 18], if the connection cost is symmetric and satisfies the triangle inequality. For arbitrary cost, the worst case bound is $O(\log n)$. However, neither of the problems can be applied directly to the design of overlay networks, since in the overlay model the communication channels between clients and servers are over the commodity Internet and do not incur any cost to service providers. Rather, the major cost is the number of servers needed to service all the clients and the access bandwidth required at each server's network interface.

Our model of server placement resembles more closely to the set cover problem. The classic greedy algorithm for solving set cover problem [10, 12] achieves an $O(\log n)$ performance ratio. In geometric spaces, the problem is easier. In [7], Hochbaum proposed a shifting strategy that gives an $(1 + \varepsilon)$ performance ratio. Unfortunately, the interconnections between networks dictate that the network propagation delay no longer exhibits the geometric properties of distance.

References [13, 14] studied the problem of placing cache replicas in the network and formulated it as the k-median problem: given a specific number of servers, what is the best placement that achieves the highest average service level to clients, where service level is indicated by access delay from a client to its nearest replica. In [13], Qiu et al. proposed several placement strategies including: a greedy strategy that incrementally places replicas to achieve highest service quality; a hot-spot strategy that places replicas near the clients that generate the greatest load. In [14], the authors also proposed a max degree strategy by placing replicas in decreasing order of nodes' degrees. By simulating over several synthetic and real network graphs, they concluded that the greedy strategy performs remarkably well, achieving within 1.1 to 1.5 of the lower bound.

Our approach to network design is from a different angle. We are more interested in examining the necessary cost, in

this case the number of servers, if we want to provide all clients a guaranteed service. This gives service providers insight into the relation of network cost and the achievable service quality, on which they can make further adjustment to reflect their revenue stream, such as eliminating servers that only serve small numbers of clients. Contrarily, the work in [13,14] seeks to optimize the average service quality which masks the number of unsatisfied customers. Additionally, the performance of our approach, which is the number of required servers, is not susceptible to the cost metric of connection paths, since we only use it to categorize clients as serviceable or not by a server; while theirs is achieved for a specific cost metric, namely the access delay. Since the connection cost metric depends heavily on the application, it is questionable if the same ratio could be achieved with a different metric.

Another difference is that we model the network geographically and consider server co-locations. As networks overlap geographically, the number of potential server locations is much fewer in number than the number of network nodes need to be considered. In [13, 14], they used network graphs consisting of tens of thousands nodes for router-level graphs and thousands of nodes for AS-level graphs. Consequently, the optimal algorithm based on LP relaxation is too expensive for their models. We think considering the geographical locations of servers is a reasonable approach given the vast presences of co-location providers. The reduced problem size enables us to solve it more optimally. In Section 5, we compare the performance of our algorithm both with co-location and without, and show that with co-location we can reduce the number of required servers to approximately half of that with no network co-locations.

3 NETWORK MODELS

We model the networks using two types of graphs: random graphs and geographic graphs. The latter consists of network nodes located at the 50 largest US metropolitan areas. For inter-domain network connectivities, we specify a set of parameters to determine the location and density of network peering points. For intra-domain network connectivities, as ISPs are not willing to disclose fully their network topology, we assume that they are able to engineer and operate their own networks with little or no congestion internally so that the delays between the routers are dominated by the link propagation delay. Consequently, we model the intra-domain network as a complete graph. We assume the "hot-potato" routing policy at the inter-domain level, which minimizes the number of network domains crossed. Hence, traffic destined to another domain is always sent to the nearest peering points from the originator towards the destination domain. Although such policy does not result in the best global routes, it is widely used by the current inter-domain routing protocol: the Border Gateway Protocol (BGP) [16]. We detail our parameter choices for the two models below and summarize the parameters in Table 1.

Parameters	Interpretations
n	network size as # of nodes
scale	size of the network graph
N_p	probability of a city in a network
TX_p	interconnection probability between two networks
TX_{scope}	scope of a region for network interconnections
TX_{ds}	interconnection density
vicinity	maximum distance between co-located nodes

Table 1: Parameters for Generating Network Graphs

Random Graph

In the random graph model, nodes are randomly distributed over a plane of size *scale x scale*. The number of nodes in each network is uniformly drawn from the interval on *[min, max]*. We divide the plane into fixed size of regions according to the parameter TX_{scope}. Nodes in different networks are allowed to interconnect with each other only if they are in the same region; nodes in the same network are fully connected. The interconnection probability TX_p decides if a pair of networks interconnect; we choose TX_p based on the size of the two networks:

$$TX_p = \alpha e^{\beta \frac{\sqrt{n_1 n_2}}{max}}$$

where n_1 and n_2 are number of nodes in the two networks, α and β are the scale and shape parameters of the probability distribution, respectively. So, two large networks are more likely to interconnect than two smaller networks.

If two networks interconnect, we randomly select a number of regions to interconnect according to the interconnection density TX_{ds}. If there are multiple nodes from each network in the same region, we select the closest pair of nodes as peers; if a region is selected, but one of the network does not have any node in that region, we choose another region until we met the peering density criterion, or we have considered all regions. We allow co-location nodes if nodes from different networks are in a geometric vicinity of each other. A server placed at a co-location can send traffic to all these networks with no additional cost.

Geographic Graph

In the geographic model, we collect the 50 largest metropolitan areas [20] as node locations. We then divide the US continent into 5 regions: northeast, north-central, southeast, south-central and west, and categorize nodes into each region with a certain amount of overlap. Details of the categorization can be found in [17]. Unlike the random graph model where all networks share the same geometric space, the geographic model consists of two types of networks: regional networks and national networks. Each city joins the network with probability N_p: the selection of nodes for a regional network considers only nodes that belong to that region; while a national network considers all 50 cities. As before, we interconnect two networks with probability TX_p. The values of TX_p may be different depending on the types of the two networks. For example, two national networks will have $TX_p = 1$, since

they are almost always interconnected; while two regional networks are less likely to peer with each other directly but to transit through a national network. We allow interconnections only if two network nodes are in the same city and use TX_{ds} to decide the number of peering points of two networks.

4 Formal Definitions and the Algorithms

Given our network models and routing policy, we can compute a routing table for each node i and the cost of each routing path $c(i, j)$, which is the summation of hop distances along the path. For each node i, we compute a set S which includes all the nodes reachable from i within the routing distance of C. If i has co-location nodes, then the set S also includes all nodes reachable from each of these co-location nodes within distance C. Let $S_1, S_2, \ldots, S_m$ be all the sets computed. An integer programming formulation of the set cover problem is:

$$\text{Objective:} \quad \text{minimize} \sum_{j=1}^{m} x_j \tag{1}$$

$$\text{Subject to:} \quad \sum_{j=1}^{m} a_{ij} x_j \geq 1 \qquad \text{for } i = 1 \ldots n \tag{2}$$

$$x_j \in \{0, 1\}$$

where x_j is the selection variable of S_j, a_{ij} is 1 if $i \in S_j$ and 0 otherwise.

A variation of the problem is to allow one primary and one backup server to cover each node. A backup server is allowed to cover more distance than the primary server. Let $T_1, T_2, \ldots, T_m$ be all the backup sets, and $b_{ij} = 1$ if $i \in T_j$ and 0 otherwise. The objective here is still to minimize the number of selected sets but with the additional constraints of:

$$\sum_{j=1}^{m} b_{ij} x_j \geq 2 \qquad \text{for } i = 1 \ldots n \tag{3}$$

Since all nodes in the primary set are also in the backup set centered at the same server, $b_{ij} = 1$ if $a_{ij} = 1$; but a primary server cannot service the same node as a backup – the constraint in (3) ensures the selection of a different server as the backup.

4.1 LP Relaxation-based Methods

The above formulation can be approximated by first solving the LP relaxation of the problem optimally and then rounding the fractional values to integers. The LP relaxation of the problem is to allow the selection variables x_j to take fractional values between $[0, 1]$. The LP relaxation can be solved in polynomial time and the rounding can be done in $O(n)$. Reference [6] introduced a rounding algorithm which is a p-approximation algorithm, where $p = \max_i\{\sum_j a_{ij}\}$ is the maximum number of sets covering an element. Although this worst case result is not very promising, we are more interested in the average case performance. We refer to the rounding algorithm in [6] as the *fixed-rounding (FR)* algorithm:

Step 1: Solve the LP relaxation of the problem and let $\{x_j^*\}$ be the optimal solution;
Step 2: Output sets $\{S_j | x_j^* \geq \frac{1}{p}\}$.

The intermediate solution for the LP relaxation naturally provides a lower bound $= \sum_j x_j^*$ for the set cover problem, because the fractional solution is an optimal solution and the LP relaxation is a super set of the set cover problem. We will use this lower bound to evaluate the quality of the solutions produced by our algorithms.

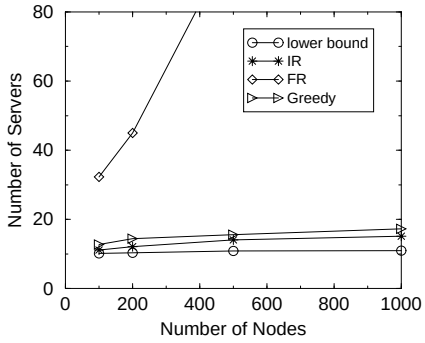
We have also devised an *incremental-rounding (IR)* algorithm that imposes more restricted rules while selecting sets based on the value of x_j^*. Whenever we select a set, we remove all the elements that satisfy the covering constraint in (2) due to the newly selected set. Let M denote the union of all elements covered after each step. For the remaining uncovered elements in a set S_j, we compute $p_j = \max_i\{\sum_j a_{ij}\}$ for $i \in S_j \backslash M$. Among all the sets that have selection variables greater than the inverse of p_j, we choose the set that has the largest number of remaining uncovered nodes.

Step 1: Solve the LP relaxation of the problem and let $\{x_j^*\}$ be the optimal solution;
Step 2: Select set S_j such that :
2(a) S_j has the largest number of uncovered elements;
2(b) $x_j^* \geq \frac{1}{p_j}$;
Step 3: Repeat step 2 until all elements are covered.

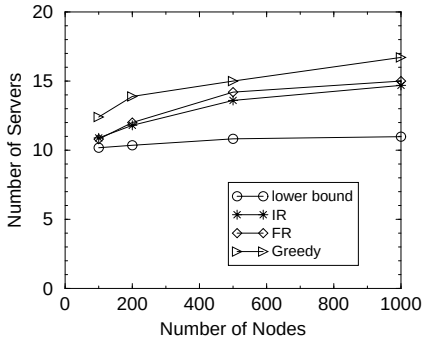
The correctness of the algorithm holds: for each uncovered node, at least one set has $x_j^* \geq \frac{1}{\sum_j a_{ij}}$ and $p_j \geq \sum_j a_{ij}$. By selecting all sets whose values satisfy 2(b), we are guaranteed to cover all the nodes. Further more, since p_j is non-increasing in each repetition and $p_j \leq p$, the set selection criterion is more restrictive than that in the FR algorithm, which in turn reduces the number of sets selected. Although the worst case bound is the same for both algorithms, we observe from our simulations that the IR algorithm typically performs much better than the FR algorithm.

An alternative to rule 2(a) is to select the set with the greatest x_j^* value, since the larger the value of the selection variable, the more "essential" the set may be. For example, if a node is covered by a single set, then the selection variable of this set must be 1 and the set must be selected. However, most of our simulations show that rule 2(a) generally performs better than this alternative rule. One plausible explanation is that rule 2(a) is more objective in attempting to include as many uncovered nodes as possible, while the alternative rule first selects those more "essential" sets, which may not contain many nodes.

It is easy to see that both of the algorithms can still have redundant sets in the final solution. To prune these extra sets, we use a simple pruning algorithm as the final step to complete the selection:



(a) Performance without the pruning routine



(b) Performance with the pruning routine

Figure 1: Comparison of the FR, IR and the Greedy Algorithms

Step 1: Sort all selected sets in increasing order of set size;
Step 2: Starting from the smallest set, check if it can be removed without leaving any of its nodes uncovered; If so, remove the set.
Step 3: Repeat Step 2 until all sets are checked.

4.2 Greedy Heuristics

A greedy algorithm is usually attractive due to its simplicity. In [10, 12], Johnson and Lovász introduced a greedy algorithm for the set cover problem with an $O(\log n)$ approximation ratio. The basic greedy attribute of the algorithm is to select a set at every step that contains the maximum number of uncovered elements. For the backup problem variant, we extend the algorithm by treating any node that has not satisfied the constraints of (2) and (3) as equally uncovered. At each step, we select a set that has the largest number of remaining uncovered nodes and repeat till there are no more uncovered nodes.

4.3 Comparison of the FR, IR and the Greedy Algorithms

We first compare our *incremental rounding (IR)* algorithm with the *fixed rounding (FR)* algorithm proposed in [6] and with the greedy algorithm. The results are further compared with the lower bound obtained as the optimal solution from the LP method. The LP solver we used, is called PCx [2] which is an interior-point based linear programming package.

We use a simple setup to investigate the relative performance of these algorithms. The underlying network graph is a single graph of randomly distributed nodes on a 100 by 100 unit length map. The service range of a server is 20 units. Ideally, if nodes are perfectly positioned, this will give a solution of $\lceil \frac{100}{40} \rceil \times \lceil \frac{100}{40} \rceil = 9$ selected servers regardless of the node density. The lower bound we obtained is indeed not far from the ideal and stays constant with the increase of the node density as shown in Figure 1.

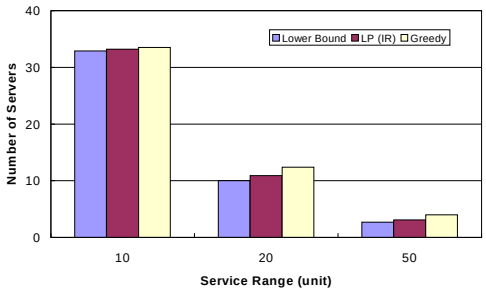
We show the performance of the rounding algorithms with and without the pruning routine in Figure 1. As expected, the FR algorithm performs badly with the increase of node density, since the number of sets covering a single node increases with node density, making the selection criterion less strict. On the other hand, the IR algorithm is always the closest to the lower bound. The FR algorithm does benefit greatly from the pruning routine, achieving performance closer to the lower bound, and is only slightly worse than the IR algorithm, but better than the greedy algorithm. This relative performance holds for other settings we have tried as well. In the rest of the paper, we will mainly focus on the IR algorithm to evaluate the placement methods in more complicated network configurations.

5 Simulation Results

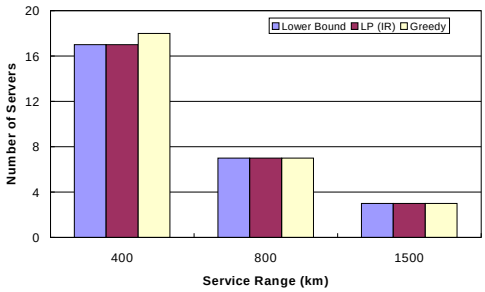
It is challenging to select a representative set of simulations that demonstrate the relationships among the methodologies, the configurations and the results, given the vast number of parameters. In order to concentrate on a few aspects which we considered interesting, we have mostly used small and uniform settings in the simulations presented in this section. We do not claim our network models capture all the fundamental characteristics of the Internet, but we believe that the combination of the random networks and the geographic networks provides a wide enough spectrum to give us some confidence in the general utility of the methods. Throughout the section, readers are referred to Table 1 for the definitions of the parameters. Unless otherwise mentioned, we use the following default parameter values. For the random graph: the scale is 100 by 100 units; the region size is 10 units and the co-location vicinity is two units; $\alpha = 0.36$ and $\beta = 1.0$; the number of nodes are chosen on the interval $[20, 100]$. For the geographic graph: the probability of including cities in the networks is 0.6 for the regional networks and 0.8 for national networks.

5.1 Single Network

We first present results on a single network for both the random graph and the geographical graph. We perform simulations on the following three scenarios: (a) $k = 1$, with only one primary server required to cover each node; (b) $k = 1$ with one backup server; (c) $k = 1$ with relaxation on the server to client distance. The last scenario allows a compromise on the service standard for a limited number of clients. This allows service providers to be more cost effective and not to install servers just for a few remotely located nodes.

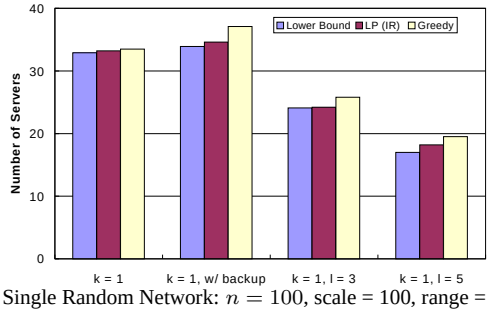


(a) Single Random Network: $n = 100$

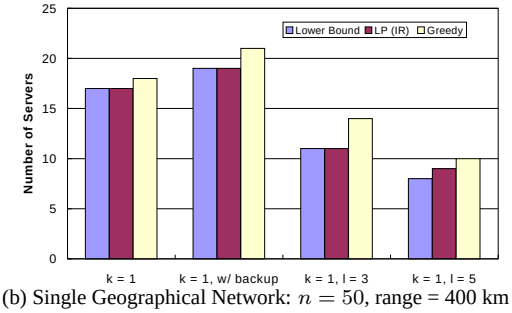


(b) Single Geographical Network: $n = 50$

Figure 2: Variation on Server Service Range



(a) Single Random Network: $n = 100$, scale = 100, range = 10



(b) Single Geographical Network: $n = 50$, range = 400 km

Figure 3: Variation on Different Service Requirement

The relaxation is done by including every node in at least l server sets, even if it is not within the range of l servers. That is, if a node is in the range of $l' < l$ servers, we add it to the sets for the next $l - l'$ servers that it is closest to. Each result for the random graph is averaged over 10 runs, while the result for the geographic graph is from a single run, since the node locations are all fixed.

Figure 2 shows the number of required servers when varying service range. In general, both the IR and the greedy algorithm closely track the lower bound.

Figure 3 shows the performance against the different service requirements. The backup server range is twice that of the primary server for both networks. It shows that the addition of the backup servers only increases the number of total servers slightly. The service range relaxation is also effective in reducing number of servers to about 75% and 55% of the original number, with $l = 3$ and $l = 5$ respectively. However, the relaxation is useful only when the service range is small. In Figure 3, the service range is 10 units and 400 km in the network configurations, which covers about 7% and 8% of the maximum distance in their respective networks. If we double the service range, we find that the relaxation becomes irrelevant since each node is likely to be included in multiple sets already. Table 2 shows the average node to server distance with and without the service range relaxation. Since there may be multiple servers covering a node, we select the closest server when computing the distance. Each result for the random graph is the worst case among 10 runs.

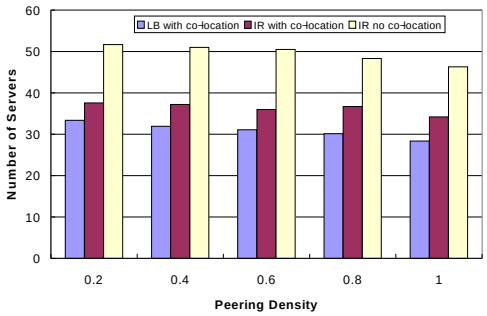
	Random Graph (unit) range = 10			Geographic Graph (km) range = 400 km		
	mean	std.	max	mean	std.	max
$l = 0$	4.94	3.74	9.83	167.49	140.11	398.79
$l = 3$	6.30	4.47	14.82	259.78	222.65	1013.42
$l = 5$	8.45	5.56	25.36	304.69	238.59	1114.78

Table 2: Average Client to Server distance

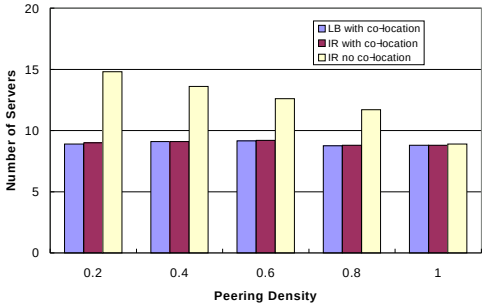
5.2 Multiple Networks

In this section, we study the relationships between server placement and the density of network peering links. By "peering links", we mean both the peering and transit relationship between two ISPs. As these links aggregate and transport traffic from one domain to another, their limited capacities contribute significantly to the user experienced network congestion. Additionally, these network exchange points maybe located off the optimal path, resulting in longer and more circuitous routes. One way to circumvent these congestion points is to use co-location services, where servers can access multiple networks and can route traffic directly to these networks without going through the exchange points. We demonstrate the relative performance with and without server co-location in Figure 4.

For this simulation, we use two network configurations: one is constructed from 5 random graphs, the other is constructed from 5 regional networks and 1 national network in the geographic model. Hereafter, we will use the term *m-n* to denote geographic networks consisting of m regional networks and n national networks. We use $TX_p = 1$, so every



(a) Random Network: 5 networks, 100 nodes per network, range = 20 units



(b) Geographic Networks: 5 regional networks, 1 national networks, total nodes = 96, range = 800 km

Figure 4: Variation on Network Peering Density

Peering Density	Random Network			Geographic Network		
	5 networks, 100 nodes per network			5 regional, 1 national network, total 96 nodes		
	total links	co-location	no co-location	total links	co-location	no co-location
0.2	83.05	22.16%	38.77%	11.6	17.24%	80.17%
0.4	158.05	23.28%	37.52%	18.9	10.05%	82.54%
0.6	238.60	21.42%	38.98%	27.4	6.20%	82.48%
0.8	317.75	22.19%	35.22%	36.3	8.82%	90.36%
1.0	410.1	21.24%	35.94%	45.2	7.30%	96.90%

Table 3: Number of Peering Links In Use

pair of networks is always interconnected, unless they do not have any common presences in any of the peering regions. The peering density in Figure 4 determines the number of peering points of two networks.

Figure 4 shows the number of required servers for the lower bound with co-location and the IR algorithm with and without co-location. The co-location service reduces the number of required servers by as much as 50% when peering is sparse. The geographic graph appears to be more sensitive to the peering density, as the performance of IR with no co-location approaches the lower bound when the peering density approaches 1. This is because the peering link is "cheaper" in a geographic network than in a random network. In the geographic network, two networks peer only if they both have presences in the same metropolitan area, which means the delay on the peering link is 0. On the other hand, the peering link has a positive delay in the random graphs which adds to the server to client delay. Additionally, the "hot-potato" routing policy always selects the closest peering link rather than the one with the lowest delay. These combined effects shows that unless peering can guarantee a high level of quality, merely increase the peering density does not help reducing the client access delay. Table 3 summarizes the number of links used in the two scenarios.

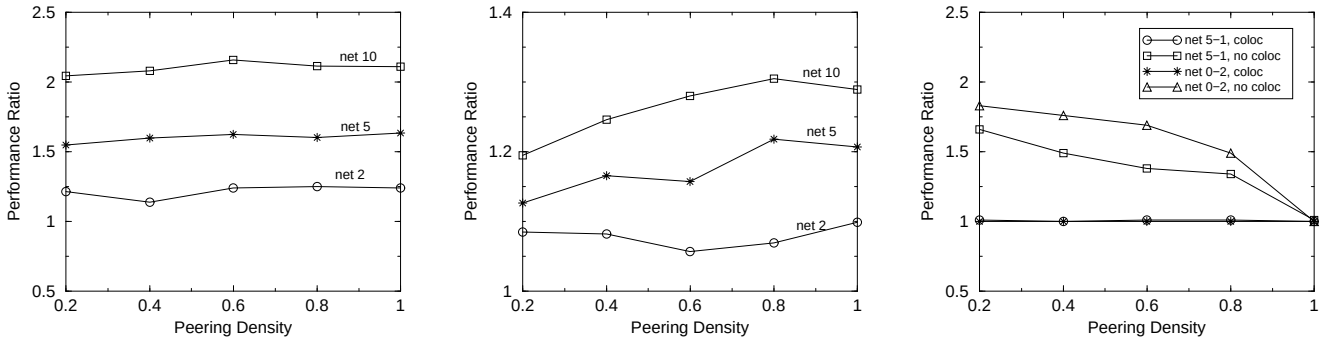
Figure 5 shows the relative performance ratio of the IR algorithm, with and without co-location, against the lower bound. We varied the number of random networks as 2, 5 and 10 and used two configurations for the geographic networks: 5-1 and 0-2. The results are mostly consistent with that in Figure 4. Additionally, it suggests that on the random networks, the performance of non co-located servers worsens much more with the increase of the number of networks, as compared to the performance of co-located servers.

Due to limited space, we have omitted the results that measure the average and the variance of server loads with different placement strategies, and the results on the computational complexity of the IR algorithm, compared with the greedy algorithm. Interested readers can find these results in [17].

6 CONCLUSIONS

We have presented a server placement method in overlay networks as an application of the set cover problem. The placement strategy satisfies constraints on the server to client paths, which indicate the obtainable service qualities along the paths. We expect that network provisioning for quality of service becomes more common as the Internet continues to grow; and such an automated methodology is useful for service providers to analyze the potential cost of network provisioning.

We solved the set cover problem using methods based on linear programming relaxation as well as greedy heuristics. We also presented an incremental integer rounding algorithm for the LP-relaxation based method. Our network settings model explicitly the presence of co-location services, which have become increasingly popular for business corporations to out source their data servers. Our results indicate co-location can save up to 50% of the server installation cost. We also presented variance of the simple set cover problem to allow backup servers and to allow distance relaxation. These variances bring opportunities to provide more cost effective services. Using simulation, we studied the behavior of the algorithms under various network settings and observed the



(a) Random Network without co-location: range = 20, 100 nodes per network.

(b) Random Network with co-location: range = 20, 100 nodes per network.

(c) Geographic Networks: range = 800 km.

Figure 5: Relative Performance Ratio Against Lower Bound

implication of network peering density and the characteristics of server load distributions. Although, LP-relaxation based methods are traditionally considered as too expensive and complex to solve any practical problems, we find that it is suitable and effective for our overlay network models and results in better performance than the greedy algorithm.

REFERENCES

[1] T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to Algorithms*. MIT Press, 1990.

[2] J. Czyzyk, S. Mehrotra, M. Wagner, and S. Wright. PCx User Guide, http://www-fp.mcs.anl.gov/otc/Tools/PCx/.

[3] M. B. Doar. A Better Model For Generating Test Networks. In *Proc. of Globecom'96*, Novemeber 1996.

[4] U. Feige. A Threshold of $O(\ln n)$ for Approximating Set Cover. In *Proc. of the Twenty-Eighth Annual ACM Symposium on the Theory of Computing*, pages 314–318, Philadelphia, Pennsylvania, May 1996.

[5] S. Guha and S. Khuller. Greedy strikes back: Improved Facility Location Algorithms. In *Proc. of the 9th Annual ACM-SIAM Symposium on Discrete Algorithms*, 1998.

[6] D. Hochbaum. *Approximation Algorithms for NP-Hard Problems*. Brooks/Cole Publishing Co., 1996.

[7] D. S. HochBaum and W. Maass. Approximation Schemes for Covering and Packing Problems in Image Processing and VLSI. *Journal of the Association for Computing Machinery*, 32(1):130–136, January 1985.

[8] iBeam Broadcasting Corp. http://www.ibeam.com.

[9] K. Jain and V. Vazirani. Primal-dual Approximation Algorithms for Metric Facility Location and k-median Problems. In *Proc. of the 40th IEEE Symposium on Foundations of Computer Science*, 1999.

[10] D. S. Johnson. Approximation Algorithms for Combinatorial Problems. *Journal of Computer and System Sciences*, 9:256–278, 1974.

[11] R. M. Karp. Reducibility Among Combinatorial Problems. *Complexity of Computer Computations*, pages 85–103, 1972.

[12] L. Lovása. On the Ratio of Optimal Integral and Fractional Covers. *Discrete Mathematics*, 13:383–390, 1975.

[13] L. Qiu, V. N. Padmanabhan, and G. M. Voelker. On the Placement of Web Server Replicas. In *Proc. of IEEE INFOCOM*, 2001.

[14] P. Radoslavov, R. Govindan, and D. Estrin. Topology-Informed Internet Replica Placement. In *Proc. of Sixth International Workshop on Web Caching and Content Distribution (WCW'01)*, June 2001.

[15] R. Raz and S. Safra. A Sub-Constant Error-Probability Low-Degree Test, and a Sub-Constant Error-Probability PCP Characterization of NP. In *ACM Symposium on Theory of Computing*, pages 475–484, 1997.

[16] Y. Rekhter and T. Li. A Border Gateway Protocol 4 (BGP-4). Internet Engineering Task Force, RFC 1771, March 1995.

[17] S. Shi and J. Turner. Placing Servers in Overlay Networks. Technical Report WUCS-02-05, Washington University, 2002.

[18] D. B. Shmoys, Éva Tardos, and K. Aardal. Approximation Algorithms for Facility Location Problems. In *Proc. of the 29th ACM Symposium on Theory of Computing*, 1997.

[19] Akamai Technologies, Inc. http://www.akamai.com.

[20] U.S. Census Bureau. http://www.census.gov/population/www/estimates/metropop.html.

[21] E. W. Zegura, K. Calvert, and S. Bhattacharjee. How to Model an Internetwork. In *Proc. of IEEE INFOCOM*, San Francisco, CA, 1996.